**13th ICCRTS**
**"C2 for Complex Endeavors"**

**Title of Paper:**

# Beyond Reactive Planning: Self Adaptive Software and Self Modeling Software in Predictive Deliberation Management

**Topics**
**C2 Architectures: Cognitive and Social Issues; C2 Concepts, Theory, and Policy**

**Authors: Jack Lenahan, Mike Nash, and Phil Charles**
**POC: Jack Lenahan**
**Organization: Office of the Chief Engineer**
**Space and Naval Warfare Systems Command**
**Charleston, S.C.**
**Address: P.O. Box 190022**
**N. Charleston, South Carolina: 29419**
**Phone: 843-218-6080**
**Email: John.Lenahan@Navy.mil**

# Report Documentation Page

*Form Approved*
*OMB No. 0704-0188*



| 1. REPORT DATE | 2. REPORT TYPE | 3. DATES COVERED |
|---|---|---|
| **JUN 2008** | | **00-00-2008 to 00-00-2008** |

| 4. TITLE AND SUBTITLE | 5a. CONTRACT NUMBER |
|---|---|
| **Beyond Reactive Planning: Self Adaptive Software and Self Modeling Software in Predictive Deliberation Management** | 5b. GRANT NUMBER |
| | 5c. PROGRAM ELEMENT NUMBER |
| 6. AUTHOR(S) | 5d. PROJECT NUMBER |
| | 5e. TASK NUMBER |
| | 5f. WORK UNIT NUMBER |
| 7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **Space and Naval Warfare Systems Command,P.O. Box 190022,N. Charleston,SC,29419** | 8. PERFORMING ORGANIZATION REPORT NUMBER |
| 9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) | 10. SPONSOR/MONITOR'S ACRONYM(S) |
| | 11. SPONSOR/MONITOR'S REPORT NUMBER(S) |

12. DISTRIBUTION/AVAILABILITY STATEMENT
**Approved for public release; distribution unlimited**

13. SUPPLEMENTARY NOTES
**13th International Command and Control Research and Technology Symposia (ICCRTS 2008), 17-19 Jun 2008, Seattle, WA**

14. ABSTRACT
**The purpose of this paper is to examine an approach to planning which extends beyond the traditional reactive planning state space. We present the following hypothesis: predictive deliberation management using self adapting and self modeling software will be required to provide mission planning adjustments after the start of a mission. Self adaptive software evaluates its own behavior and changes behavior when the evaluation indicates that it is not accomplishing what the software is intended to do, or when better functionality or performance is possible1. Self modeling systems construct their own abstractions as a basis of computational intelligence2. In order to provide a proper process context for the evolution of software toward this level of autonomy, and in alignment with the proposed planning maturity models3, we put forth a concept of a NCW C2 Software Maturity Model. This new C2 software maturity model will take software beyond the service oriented paradigm into a new era of software designing its own replacements or modifications in order to satisfy new command and control requirements.**

15. SUBJECT TERMS

| 16. SECURITY CLASSIFICATION OF: | | | 17. LIMITATION OF ABSTRACT | 18. NUMBER OF PAGES | 19a. NAME OF RESPONSIBLE PERSON |
|---|---|---|---|---|---|
| a. REPORT **unclassified** | b. ABSTRACT **unclassified** | c. THIS PAGE **unclassified** | **Same as Report (SAR)** | **26** | |

**Beyond Reactive Planning: Self Adaptive Software and Self Modeling Software in Predictive Deliberation Management**

Authors: Jack Lenahan, Mike Nash, Phil Charles

**Abstract**

The purpose of this paper is to examine an approach to planning which extends beyond the traditional reactive planning state space. We present the following hypothesis: predictive deliberation management using self adapting and self modeling software will be required to provide mission planning adjustments after the start of a mission.

Self adaptive software evaluates its own behavior and changes behavior when the evaluation indicates that it is not accomplishing what the software is intended to do, or when better functionality or performance is possible[1]. Self modeling systems construct their own abstractions as a basis of computational intelligence[2].

In order to provide a proper process context for the evolution of software toward this level of autonomy, and in alignment with the proposed planning maturity models[3], we put forth a concept of a NCW C2 Software Maturity Model. This new C2 software maturity model will take software beyond the service oriented paradigm into a new era of software designing its own replacements or modifications in order to satisfy new command and control requirements.

## Introduction

In this paper we explore the possibility of expanding the definition of predictive deliberation to include computing the time necessary for permitting self adaptation to solve planning problems after the plan execution begins. We discuss several agent architectures to accomplish this based upon the notion of agent reputation. Through self modeling, an "honest agent" will be able to self evaluate whether or not it was capable of producing the changes required to successfully adjust a plan. If the original un-adapted agent believed that it could make the necessary changes to the plan, it would simply "adapt the plan" to the new requirements. If the honest agent did not believe that it had the planning capability required, it would adapt itself. We advance the following approach to self-adaptation:

1. Determine the new capabilities required and then find the new planning capabilities required by conducting a "reputation based search" on the GIG. After discovering the required components and services, the agent will then bind to them thus creating a new version of itself capable of solving the new planning problems in the time allocated by the "predictive deliberator function". If the merging of capabilities results in a set of integrated agents under the control of a manager agent rather than a single large or complex agent, I have dubbed this configuration a "Mogul".
2. The reputation mechanism we propose is a modified aggregation operator necessary to perform dynamic reputation scoring possibly from many sources concerning the same agent or collection of agents acting as a mogul.

Finally we propose a C2 agent planning software capability maturity model which would hopefully provide a roadmap for the creation of this level of capability.

## Self Modeling Systems

The literature describes "Self modeling systems" as systems capable of constructing their own goals rather than simply attempting to carryout goals given to it exclusively from the outside[4.] Reactive agents are too slow to operate effectively in complex environments; we need anticipatory agents with models of their own processes so they can identify anomalies in their behavior or in their environment's behavior[5]. Suppose a planning agent fails to successfully plan for a particular task or set of tasks, it is the goal of this type of autonomous system to maintain rules which say "I can do better than this and I need to improve by adapting because my current self model does not demonstrate sufficient capability to perform a particular task successfully" (source: footnote citation 2 above).

Please note that this can be enhanced by continuous synthetic stimulation, adaptation, and testing when the agent is not engaged in a mission. The agent can then anticipate future capability needs for itself and include these in its self model. Thus, the agent can begin the process of self adaptation without the pressure of time constraints.

## Predictive Deliberation and Self Adaptation

Predictive deliberation attempts to predict in advance how many remaining decisions there are to make, how much time there is to make a particular decision and how long it will take a particular

planning agent to make that decision. A key point of this paper is to recommend adding the following features to predictive deliberation:

1. An agent planner competency test
2. An adaptation estimation capability which will determine if there is enough time for the agent to adapt itself into a stronger planner.

For example if we assume the perspective of a planning agent in this context, we may have the following conversation with ourselves as planning agents: "I may need to evaluate my own planning competency rating or reputation since my planning reputation does not match the planning skills or planning competency required by the change in the mission. But if I had two additional components inside me, I could accomplish the planning task in half the required time." In other words, extend the predictive deliberation management functions to ask the following question: "what capabilities would an agent need to perform this task in time and is it possible to reconfigure (adapt) the existing planning agents in time to complete the superior plan?"

The need to change plans successfully and dynamically is always present in military operations. To be overwhelmed by data or to be overwhelmed by the need to produce decisions in near real time imposes severe time management constraints on agent based planners. But what if instead of traditional re-planning of alternatives by the same system or set of agents, I ask a different question:
*Am I good enough at planning?* And if my (agent) planning skills do not match the planning tasks to be accomplished in the time predicted, do I have enough time to reconfigure myself into a better planner? (Predictive deliberation with self adaptation of the agents)
By requesting services from other agents, the incomplete agent could integrate the components within its own code body or establish a collaborative planning capability with other agents with the newly required skills.
Several schemes are available:

1. The component centric work of Karsai[6] et.al involves a technique which includes a supervisory layer and a run time infrastructure which does not use CORBA style interfaces.
2. The next type of agent planning model which I have termed a Mogul, is based upon Minsky's[7] work and in effect becomes a "society of planning agents", each agent providing a specific planning capability invoked as required by a supervisory agent or Mogul. It is the Mogul's task to engage in contract and QoS negotiations with the other agents required for a particular task and to establish the agent based communications and data strategy required to be successful with such an architecture. The predictive deliberator in this case must have predicted that sufficient time exists for agent architecture creation on the fly, contract negotiations, and architectural testing.
3. The "Huge Local Component Library" concept usually is applied to traditional software applications. Everything in the library is put in the include file along with an enormous set of "hard wired rules" which invoke the different modules upon the demand of some supervisory component, recompiling only the required modules at run time.
4. Probabilistic agent architectures which, given a short time to reconfigure by the predictive deliberator, compute a probabilistic model of which other components or agents will be likely satisfy a given set of new requirements and then construct the probabilistic model under the assumption that the new configuration will actually generate an acceptable plan and then execute the plan at run time with no testing.

5. Workflow[8] or BPEL based architectures, which may consist of agents, components, web services, or any combination of these structures. Architectures of this type are less inviting since they usually rely only on web service descriptions (UDDI level descriptions), or component descriptions or other types of Meta-data which do not support the rather detailed level of data modeling required to perform dynamic reconfigurability. The other issue with this approach is that the BPEL and Workflow engines are usually human centric and therefore are incompatible with dynamic reconfigurability.

**How do I know that a given planning agent is better than another?**

This paper recommends that we use the notion of reputation as developed by e-Bay for trading. Given a well defined planning task, agents that have performed that task to a given skill level in the past, will be given a "planning competency rating" or "planning reputation" based upon the use of a modified aggregation operator scheme. The aggregation operators will be managed by a reputation service provider and maintained as repository data in the reputation granting and scoring system. Also the reputation data can be maintained as agent Meta data residing in an internal model of the planning agent itself.

**Reputation**

Reputation is the process and the effect of transmission of a target image[9]. More precisely, reputation is defined to consist of three distinct but interrelated objects: (1) a cognitive representation, or more precisely a believed evaluation; (2) a population object, i.e., a propagating believed evaluation; and (3) an objective emergent property at the agent level, i.e., what the agent is believed to be. In fact, reputation is a highly dynamic phenomenon in two distinct senses: it is subject to change, especially as an effect of corruption, errors, deception, etc.; and it emerges as an effect of a multi-level bidirectional process. In particular, it proceeds from the level of individual cognition to the level of social propagation (**bottom-up**) and from this level back to that of individual cognition again (**top-down**).

Reputation is known to be a ubiquitous, spontaneous and highly efficient mechanism of social control in natural societies. It is a subject of study in social, management and technological sciences. Its influence ranges from competitive settings, like markets, to cooperative ones, like firms, organizations, institutions and communities. Furthermore, reputation acts on different levels of agency, individual and supra-individual. At the supra-individual level, it concerns groups, communities, collectives and abstract social entities (such as firms, corporations, organizations, countries, cultures and even civilizations). It affects phenomena of different scale, from everyday life to relationships between nations. Reputation is a fundamental instrument of social order, based upon distributed, spontaneous social control. Reputations, in many instances are maintained or estimated by aggregation operators. In various applications where information fusion or multi-factorial evaluation is needed, an aggregation process is carried out… For instance, in a multi-person, multi-aspect decision problem, each alternative is evaluated by a matrix of ratings where the rows represent evaluations by persons and the columns represent evaluation by criteria. One may, for each row, merge the ratings according to each column by some aggregation operation A and form as such a global rating of each person, and then merge the person's opinions using another aggregation operation B[10]. The resulting score then becomes

the basis of the reputation. This permits dynamic reputation management of fluctuating behavior of planning agents (whether individual or a group of agents acting under the control of a mogul) and also permits granular scoring for lower level planning functions since a different set of aggregation operators can be selected under certain conditions. For a good discussion of aggregation operators, please see "Aggregation Operators and Commuting, Susanne Saminger-Platz et. al, in IEEE Transactions on Fuzzy Systems, December 2007, Volume 15, Number 6".

**C2 Planning Agent Software Maturity Model Recommendation**

We are proposing that the following model be considered by the C2 community as a basis of moving software development towards self adapting software in support of dynamically adaptive planning functions. It should be noted immediately that this recommendation is not a modification of the CMMI model as this paper is not concerned with the software development process itself. What we are concerned about is the genre of software developed not how the software will be written. We would like software planning agents to co-evolve in accordance with the Planning Maturity model developed by Alberts and Hayes[11]. It is our opinion that planning systems that behave under classical software paradigms will not be capable of providing the functionality required to meet the demands of Network Centric Warfare implied in the graphic below.

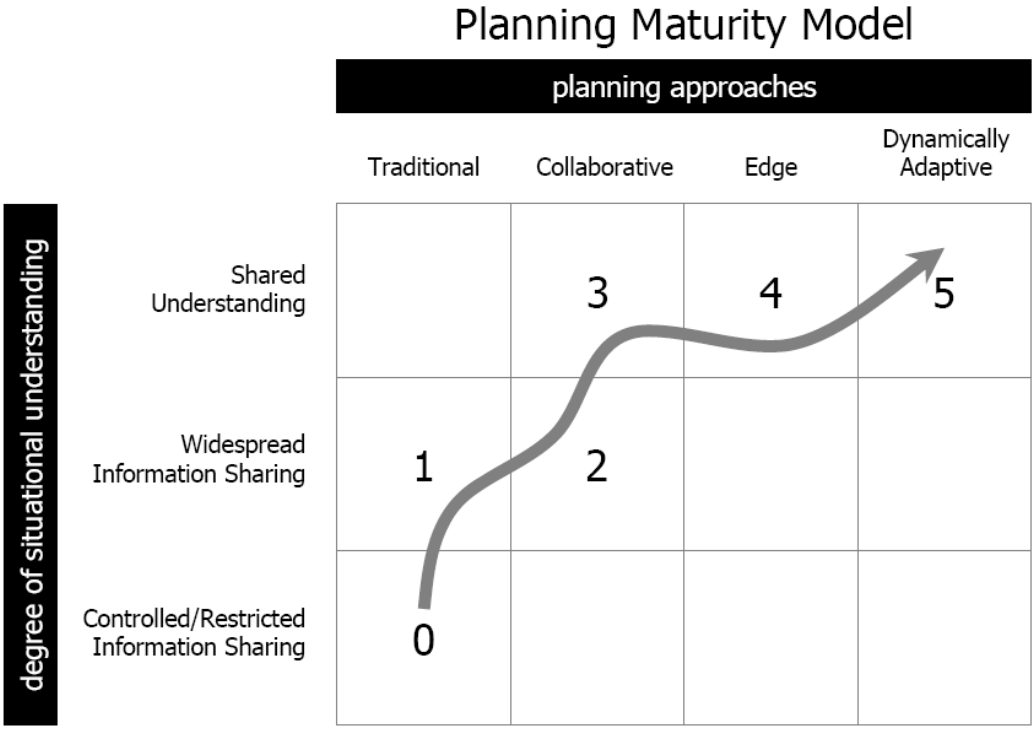



Figure 1 Planning Maturity Model

As the theme of this paper suggests, we believe that in order to achieve dynamically adaptive planning, you will need dynamically adaptive software in the form of self modeled and self adaptive software agents. To achieve this we need to move away from traditional human centric

development paradigms and begin to involve the software itself in designing its own replacements by a process of self modeling which will permit the software agent to understand that it needs to improve itself through a process of self adaptation in which new modules are combined internally to its initial structures.
Planning systems which require large and expensive "hard wired" planning algorithms whether traditional or agent based will not meet the needs of Network Centric Warfare planners as described in the Albert's paper. Collaboration between humans also currently requires expensive collaboration systems which must be locally resident. By introducing inter-agent communications we can begin to slowly move towards a greater degree of information sharing. This is already happening in the human world in terms of "Chat software". Chat is replacing many older forms of communications and resulting in better situational awareness by the users. Agents must have this ability in order to create a complex plan in a distributed manner rather than in a localized traditional manner. In order to move to the vision of edge planning, we must overcome the ever present criticism of how do you know whether or not a given "edge planner" is any good at planning a particular task? The notion of "reputation" seems to address this issue. Obviously, humans have relied upon the concept of reputation for centuries. You always want to buy your meat from the butcher with the best reputation or have your children taught by the teachers with the best reputation, so it is not at all surprising that e-bay went to this clever methodology to support its trust requirement for auctions. By moving the concept of "reputation" to agent based planning we believe that the agent based planner can actually support the vision of decentralizing command and control by pushing decision rights to the "edge".

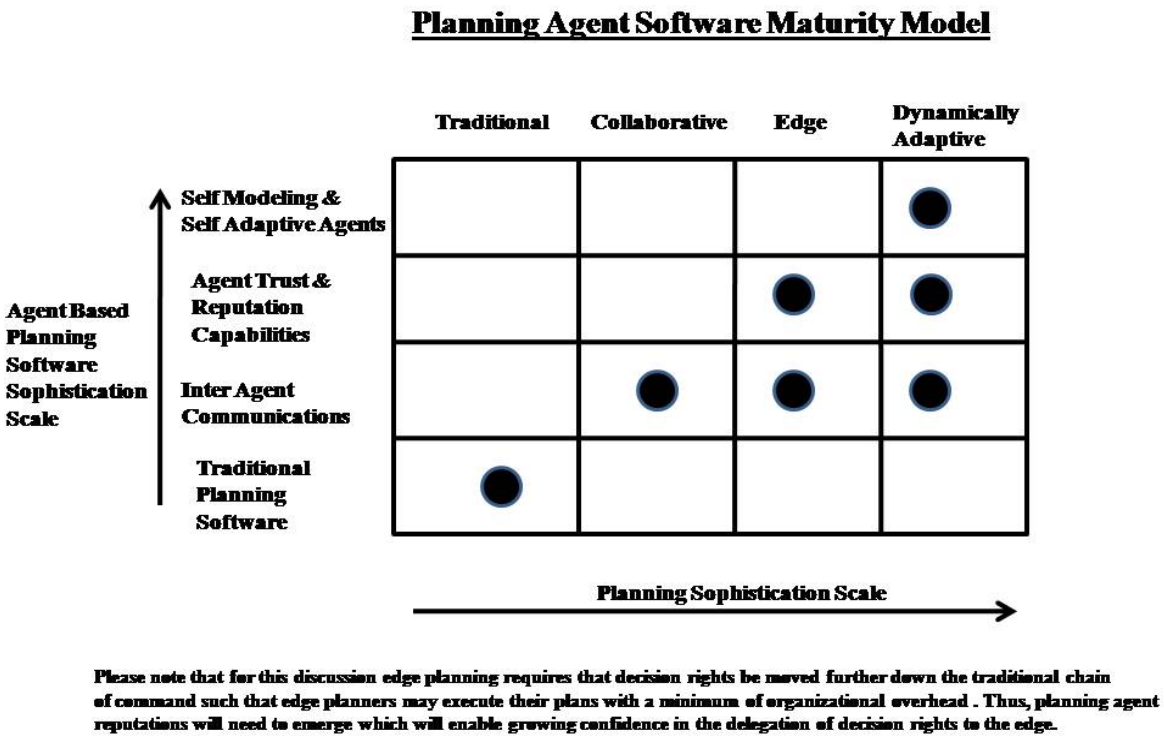



**Figure2 Planning Software Agent Capability Maturity Model**

The diagram above depicts the proposed agent based planning software evolutionary cycle. The y axis depicts agent planning software sophistication, while the x axis depicts the sophistication in the type of plans created. Legacy planning systems, at this time, are completely incapable of changing their own software; by this we mean that they are not dynamically adaptive. Distributed systems, which may support collaboration, improve planning quality by permitting a more diverse set of planners or a more experienced set of planners to contribute to the final product. For agent based systems, this requires agent to agent communications at the task level to permit the required granularity to evolve. Edge planning agents represent a different category of sophistication in that they require a co-evolutionary development in organization and process to synchronously emerge in the military chain of command strategy. Agents cannot authorize themselves to allocate decision rights to the edge. Pushing decisions to traditionally lower ranks requires some confidence that the "new decision makers" will be successful. This will require that the institutional bias against data sharing be mitigated at the doctrinal level. It also will require that the training and testing of "edge planners" be much more robust. By allocating decisions to the edge, we enable the next phase of maturity: dynamically adaptive planning capabilities. Please note that this will require that a "trust" capability and rudimentary "reputation" capability be made available to call as a service suite. SOA[12] Trust is defined as: Trust of party A in a party B for a service X is the measurable belief of A in B behaving dependably for a specified period within a specified context in relation to X.
The conventional thinking with regard to software believes that an SOA, with BPEL and Trust capabilities is the only methodology capable of achieving this result. I believe that this is too narrow a view of software possibilities. By including "self modeling" and "self adaptive" agents, whether defined as services or components, I believe that we have a greater chance of actually achieving these goals than by simply focusing on a pure SOA which will require millions of dollars of code to support. By modifying reputation schemes to support planning agent reputations in a military context, we should be able to support dynamically adaptive (self modeling and self adapting agents) in a few years. The notion of reputation, while building upon trust, will almost require its own maturity model. The following diagram[13] represents the possible complexity involved in modeling reputation as attributes of services, components or agents. Please note both individual and group reputations must be supported. In fact, group reputations must not only include the reputation of each participating agent with an aggregate reputation for the whole, it must define the specific context and problem domain that the assembly of agents was operating under.

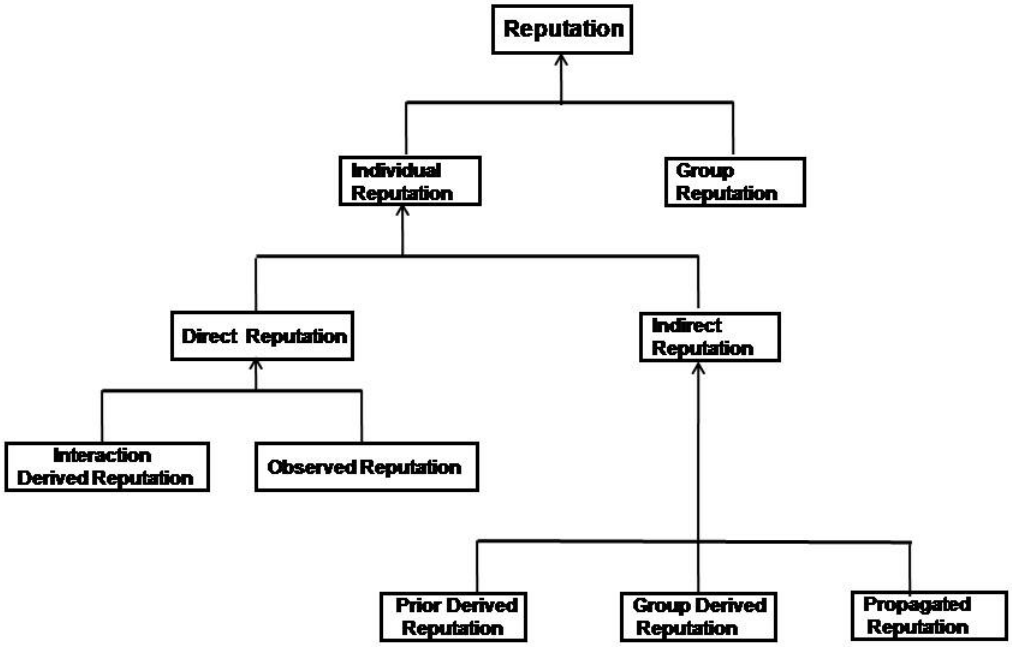



**Figure 3 – Reputation Architecture Model**

As the above model suggests, simply having a "hard wired reputation" or a simplistic aggregated reputation will be insufficient to cope with the many contexts in which in a self adapting agent will be called as a service. This model itself must adapt with usage, testing and further insights into what it means to have adapted. By this I mean that a particular agent before adaptation had a reputation say r1, but after morphing into some new configuration, it behaves such that it earns a new reputation r2. What is the meaning of the two reputations with respect to the old agent and the new agent? Suppose the agent reconfigures itself 50 times? Do we now have 50 reputations, a single new reputation or secondary reputation in that besides the agents planning reputation, it now has a reputation for successfully adapting or reconfiguring itself to new requirements? I believe that the answer to this issue is that the agent and the aggregation scoring mechanisms must maintain at least 2 reputations: the agent's planning reputation and the agent's adaptability reputation.

**Beyond BPEL and the SOA: An experiment comparing Workflow and BPEL engines versus peer to peer agent communications**

Anyone familiar with BPEL engines knows that a human being is required to create the required business processes. These business processes contain the sequences of events and steps which must be executed in a particular order. In a C2 context, these processes will most probably be mission threads. Most tools attempt to connect web services, components services, or wrapped legacy interfaces into an executable sequence of events. But if we had planning agents, assuming

that we provided these agents with a set of mission requirements in an understandable syntax, the agents could discover other agents on a mission capability reputation basis as described above, and then communicate with each other via agent based peer to peer communication strategies. Why would we need a formal BPEL engine at all as long as we comply with BPEL4WS? Research in this area has been conducted by Blanchet, et al[14]. We have simulated this concept in MySql and discovered that agents constructed as stored procedures, executing against the same database engine could indeed communicate sequentially as required in a mission thread without the need for a BPEL engine or formal workflow sequencer. Obviously this concept needs much more research. We recommend that a serious and well structured analysis of this concept be undertaken before pronouncing the death of BPEL Engines and Workflow.

**Related research on this topic[15]**:
"Conducting the execution of a process is the sequence of sending and interpreting messages that contain continuations. There are some specific issues to be addressed for peer-to-peer process execution using continuations.
1. A partially executed process must be rolled back if some fault event occurs. To enable process rollbacks, two continuations are associated with any particular execution point. The *success* continuation represents the path of execution towards the successful completion of the process. The *failure* continuation represents the path of execution towards the proper compensation of committed effects after certain failure events.
2. Some management tasks, such as proper handling of parallel branches and termination of scopes, must be carried out by the proper sites at proper time. The management tasks are defined as *auxiliary activities* that are automatically added into continuations during execution.
3. Parallel branches must be kept track of in order to: (a) stop all branches of a scope when the scope terminates, and (b) stop and rollback all branches of a scope in case of some fault event. This is done with *scope agents*. More specifically, a message for process execution contains a control activity and two continuations. The *control* activity is the activity to be executed immediately. It is either a BPEL activity or an auxiliary activity. One of the two continuations, the success or the failure continuation, is to be executed after the control activity. A continuation is represented as a stack of activities. The local architecture at a site is shown in Figure 3. Requests for process executions at the site are delivered to its message queue (1). A process interpreter is a pool of threads that interpret the messages. A thread dequeues a message from the message queue (2) and decides the next action according to the control activity of the message. There are two possibilities here: either can the process move on with local processing, or it is dependent on some other messages that are not available yet, such as a provided service waiting for an incoming invoking message. In the former case, the thread invokes (3, 4) some local programs, which might interact with human users. In the latter case, the current message is put in the pending message pool (5). This message will be used later (6) when a dependent message is available (2 again). After the execution of local programs, new messages are either put in the pending message pool (5) or sent to a remote site (7).

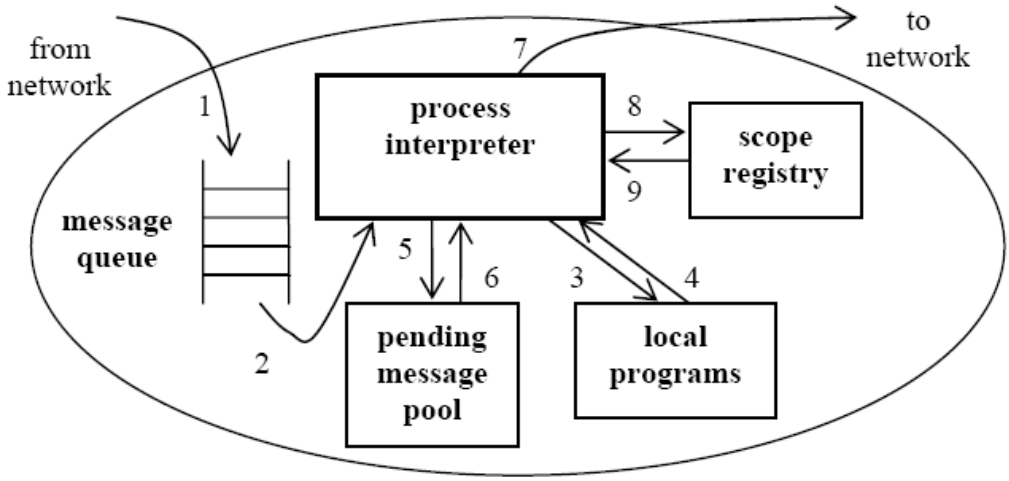



**Figure 4 Local Architecture of a Site**

If a site is also a scope agent, it maintains the scope state in the scope registry (8, 9). Basically, the scope state contains the current locations of all active parallel branches. The location of a branch changes when a message is sent. To keep this location state up to date, when a site sends a message (7), it also notifies the management agent of the immediate enclosing scope. To terminate a scope, the scope agent asks all registered sites to stop and clear up the corresponding local activities. To rollback a scope, all these sties run the respective parts of failure continuations."

## Conclusions

In order to achieve the goals of dynamically adaptive planning and predictive deliberation, we believe that a movement must be made towards software that is able to reconfigure itself on the fly. Our research indicates that self modeling and self adapting agents, which can earn a reputation, offer the best approach to solving this problem. Our research and that of others also indicates that the use of peer to peer agent communications offers a much more flexible methodology in terms of orchestrating mission sequences than does a BPEL or Workflow engine which requires human intervention.

## Summary

In this paper we have explored the possibility of expanding the definition of predictive deliberation to include computing the time necessary for permitting self adaptation to solve planning problems after the plan execution begins. We discussed several agent architectures to accomplish this based upon the notion of agent reputation. Through self modeling, an "honest agent" would be able to self evaluate whether or not it was capable of producing the changes required to successfully adjust the plan. If the original un-adapted agent believed that it could make the necessary changes to the plan to be successful, he would simply "adapt the plan" to the new requirements. If the agent (honest) did not believe that it had the planning capability required, we explored the following:

1. Search for the new planning capabilities required by conducting a "reputation based search" on the GIG. After discovering the required components and services, the agent would then bind to them thus creating a new version of itself capable of solving the new planning problems in the time allocated by the "predictive deliberator function".
2. A reputation mechanism necessary to perform dynamic reputation scoring possibly from many sources concerning the same agent or collection of agents acting as a mogul.
3. A peer to peer agent communication mechanism which may be more robust than traditional Workflow and BPEL engines.

Finally we addressed a C2 agent planning software capability model which would hopefully provide a roadmap for the creation of this level of capability. We believe that the SOA and Component based architectures must be fused into a new paradigm supported by "reputations" in order to move agent based planners beyond reactive planning.

**References**


1. Laddaga, Robert, Robertson,,Paul and , Shrobe Howie, *Introduction to Self-adaptive Software: Applications*, Self-Adaptive Software: Applications Second International Workshop, IWASAS 2001 Balatonfured, Hungary, May 2001 Revised Paper, page 1
2. Landauer, Christopher and Bellman, Kirstie L., *Self-modeling Systems*:, Self-Adaptive Software: Applications, Second International Workshop, IWASAS 2001 Balatonfured, Hungary, May 2001 Revised Papers, page 238
3. Alberts, David S. and Hayes, Richard E., *Planning: Complex Endeavors*, CCRP Press, page 161.
4. Landauer,Christopher and Bellman,Kirstie L., *Self-modeling Systems*:, Self-Adaptive Software: Applications, Second International Workshop, IWASAS 2001 Balatonfured, Hungary, May 2001 Revised Papers, page 239
5. Landauer,Christopher and Bellman,Kirstie L., *Self-modeling Systems*:, Self-Adaptive Software: Applications, Second International Workshop, IWASAS 2001 Balatonfured, Hungary, May 2001 Revised Papers, page 241
6. Karsai, Gabor, Akos Ledeczi, Janos Sztipanovits, Gabor Pecili, Gyula Simon, and Tamas Kovacshazy, An Approach to Self Adaptive Software, Based on Supervisory Control, :, Self-Adaptive Software: Applications, Second International Workshop, IWASAS 2001 Balatonfured, Hungary, May 2001 Revised Papers, page 24
7. Minsky, Marvin, "Society of Mind", TouchStone Book, Published by Simon & Shuster, 1895
8. Bose, Prasanta, Matthews, Mark,: "Dynamic Change in Workflow-Based Coordination of Distributed Services", in Self-Adaptive Software: Applications, Second International Workshop, IWASAS 2001 Balatonfured, Hungary, May 2001 Revised Papers, page 171
9. Source: http://megatron.iiia.csic.es/mediawiki/index.php?title=print_version&redirect=no#Reputation

**10.** Saminger-Platz, Susanne et. al, Aggregation Operators and Commuting, in IEEE Transactions on Fuzzy Systems, December 2007, Volume 15, Number 6".
**11.** Alberts, David S. and Hayes, Richard E., "Planning: Complex Endeavors"
**12.** Dimitrakos, Theos, "SOA Trust Management Framework", in "Trust, Reputation, and Security: Theories and Practice, AAMAS 2002 International Workshop, Bologna, Italy, July 2002, Selected and Invited papers.
**13.** Mui , Lik, Halberstadt, Ari, and Mohtashemi, Mojdeh, "Evaluating Reputation in Multi-Agent Systems", in "Trust, Reputation, and Security: Theories and Practice, AAMAS 2002 International Workshop, Bologna, Italy, July 2002, Selected and Invited papers.
**14.** Blanchet, Warren, Stroulia, Eleni, Elio , Renée, "Supporting Adaptive Web-Service Orchestration with an Agent Conversation Framework", *University of Alberta,2005*
**15.** Yu, Weihai, "Peer-to-Peer Execution of BPEL Processes", Department of Computer Science, University of Tromsø, Norway

**Biographies**

**Mr. Phil Charles** is the command chief engineer for the Navy's Space and Naval Warfare Systems Center, located in Charleston, South Carolina. His primary research interests include the operationalization of Network Centric Warfare, Command and Control Research, the use of automated battle management aids in the construction of engagement packages, and the optimization of complex end to end mission threads. Mr. Charles recent CCRP publications, co-authored with Jack Lenahan include "The Grand Challenges of Network Centric Policy", "Measuring the Effects of Cumulative Influence: Using NCW to Prevent or Minimize Civilian Causalities ", and "Assessing Self Organization and Emergence in C2 Processes".

**Mr. Jack Lenahan** is staff scientist to the Command Chief Engineer at SPAWAR Charleston and is currently employed by Imagine-One Corporation. His research interests include the application of advanced technology and software in the field on command and control, the use of evolutionary computation in the creation of engagement packages by automated battle management aids, and an analysis of Grover's and Shor's algorithms in quantum computing. Mr. Lenahan's most recent ACM publication is "The Synthesis of Evolutionary and Quantum Computing" published in the journal SIGEVOLUTION, ACM, 2006. His recent CCRP publications include "The Data Warehouse in Service Oriented Architectures and Network Centric Warfare", "Agile Assessment Techniques for Evaluating Mission Portfolio Ensembles in Complex Adaptive Architectures", and the "Is the SOA the only Valid Architectural Approach for the Transformation to NCW?"

**Mr. Mike Nash** is an industrial engineer in the chief engineer's office at SPAWAR Systems Center Charleston. His projects include research in genetic algorithms, human decision modeling, satellite and sensor simulation, and efficient design of optimization algorithms. He received a Master of Science degree from Georgia Tech in 2005 in the field of operations research. He studied industrial engineering and operations research at Cornell University, where he received a Bachelor of Science degree in 2003. While at Cornell, he received numerous awards and achievements, including the Roger Berman prize for excellence in oral presentation.

**13th ICCRTS**
**"C2 for Complex Endeavors"**
**Title of Paper:**
# Beyond Reactive Planning:
# Self Adaptive Software and Self Modeling Software in Predictive Deliberation Management

**Topics**

**C2 Concepts, Theory, and Policy: Cognitive and Social Issues**

**Authors: Jack Lenahan, Mike Nash, and Phil Charles**
**POC: Jack Lenahan**
**Organization: Office of the Chief Engineer**
**Space and Naval Warfare Systems Command**
**Charleston, S.C.**
**Address: P.O. Box 190022**
**N. Charleston, South Carolina: 29419**
**Phone: 843-218-6080**
**Email: John.Lenahan@Navy.mil**

# Introduction

- The purpose of this paper is to examine an approach to planning which extends beyond the traditional reactive planning state space. We present the following hypothesis: predictive deliberation management using self adapting and self modeling software will be required to provide mission planning adjustments after the start of a mission. This to support the requirement for dynamically adaptive planning.
- Self adaptive software evaluates its own behavior and changes behavior when the evaluation indicates that it is not accomplishing what the software is intended to do, or when better functionality or performance is possible.
- Self modeling systems construct their own abstractions as a basis of computational intelligence.
- In order to provide a proper process context for the evolution of software toward this level of autonomy, and in alignment with the proposed planning maturity models, we put forth a concept of a NCW C2 Software Maturity Model. This new C2 software maturity model will take software beyond the service oriented paradigm into a new era of software designing its own replacements or modifications in order to satisfy new command and control requirements.

# Introduction Continued

- In this paper we explore the possibility of expanding the definition of predictive deliberation to include computing the time necessary for permitting self adaptation to solve planning problems after the plan execution begins. We discuss several agent architectures to accomplish this based upon the notion of agent reputation. Through self modeling, an “honest agent” will be able to self evaluate whether or not it was capable of producing the changes required to successfully adjust a plan. If the original un-adapted agent believed that it could make the necessary changes to the plan, it would simply “adapt the plan” to the new requirements. If the honest agent did not believe that it had the planning capability required, it would adapt itself.
- We advance the following approach to self-adaptation:
  - Determine the new capabilities required and then find the new planning capabilities required by conducting a “reputation based search” on the GIG. After discovering the required components and services, the agent will then bind to them thus creating a new version of itself capable of solving the new planning problems in the time allocated by the “predictive deliberator function”.
  - If the merging of capabilities results in a set of integrated agents under the control of a manager agent rather than a single large or complex agent, we have dubbed this configuration a “Mogul”.
- The reputation mechanism we propose is a modified aggregation operator necessary to perform dynamic reputation scoring possibly from many sources concerning the same agent or collection of agents acting as a mogul.
- Finally we propose a C2 agent planning software capability maturity model which would hopefully provide a roadmap for the creation of this level of capability.

# Self Modeling Systems

- Self modeling systems construct their own abstractions as a basis of computational intelligence
- The literature describes “Self modeling systems” as systems capable of constructing their own goals rather than simply attempting to carryout goals given to it exclusively from the outside
- Reactive agents are too slow to operate effectively in complex environments
- We need anticipatory agents with models of their own processes so they can identify anomalies in their behavior or in their environment’s behavior.
- Suppose a planning agent fails to successfully plan for a particular task or set of tasks, it is the goal of this type of autonomous system to maintain rules which say “I can do better than this and I need to improve myself by adapting because my current self model does not demonstrate sufficient capability to perform a particular task successfully”
- Please note that this can be enhanced by continuous synthetic stimulation, adaptation, and testing when the agent is not engaged in a mission. The agent can then anticipate future capability needs for itself and include these in its self model. Thus, the agent can begin the process of self adaptation without the pressure of time constraints.

# Self Adapting Systems

- Self adaptive software evaluates its own behavior and changes behavior when the evaluation indicates that it is not accomplishing what the software is intended to do, or when better functionality or performance is possible[1].

# Predictive Deliberation and Self Adaptation

- Predictive deliberation attempts to predict in advance how many remaining decisions there are to make, how much time there is to make a particular decision and how long it will take a particular planning agent to make that decision.
- In addition we are modifying this definition to state that predictive deliberation must determine if an agent has the time to ‘self modify’ or adapt itself given a new planning requirement

# A Planning Agent Talks to Itself

- If we assume the perspective of a planning agent in this context, we may have the following conversation with ourselves as planning agents: “I may need to evaluate my own planning competency rating or reputation since my planning reputation does not match the planning skills or planning competency required by the change in the mission. But if I had two additional components inside me, I could accomplish the planning task in half the required time.” In other words, extend the predictive deliberation management functions to ask the following question: “what capabilities would an agent need to perform this task in time and is it possible to reconfigure (adapt) the existing planning agents in time to complete the superior plan?”

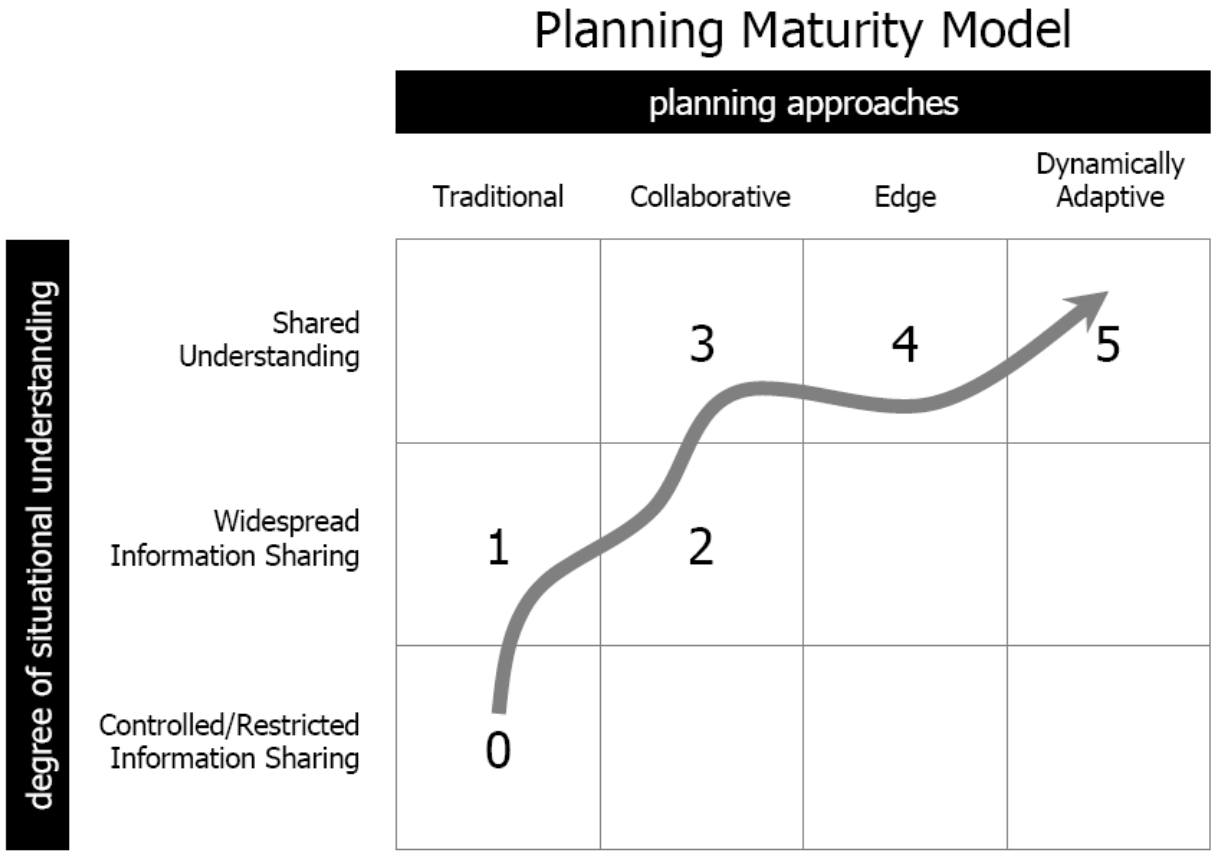
# How does the agent self adapt to new planning requirements?

- First it asks itself if it is competent to perform the new planning task requirements
- Second if the answer to the above question is no, it can take 2 approaches:
  - 'Outsource the Planning Work' work to other agents on a reputation bidding basis
  - Discover other agents through a reputation based search and bind to them forming a larger version of itself

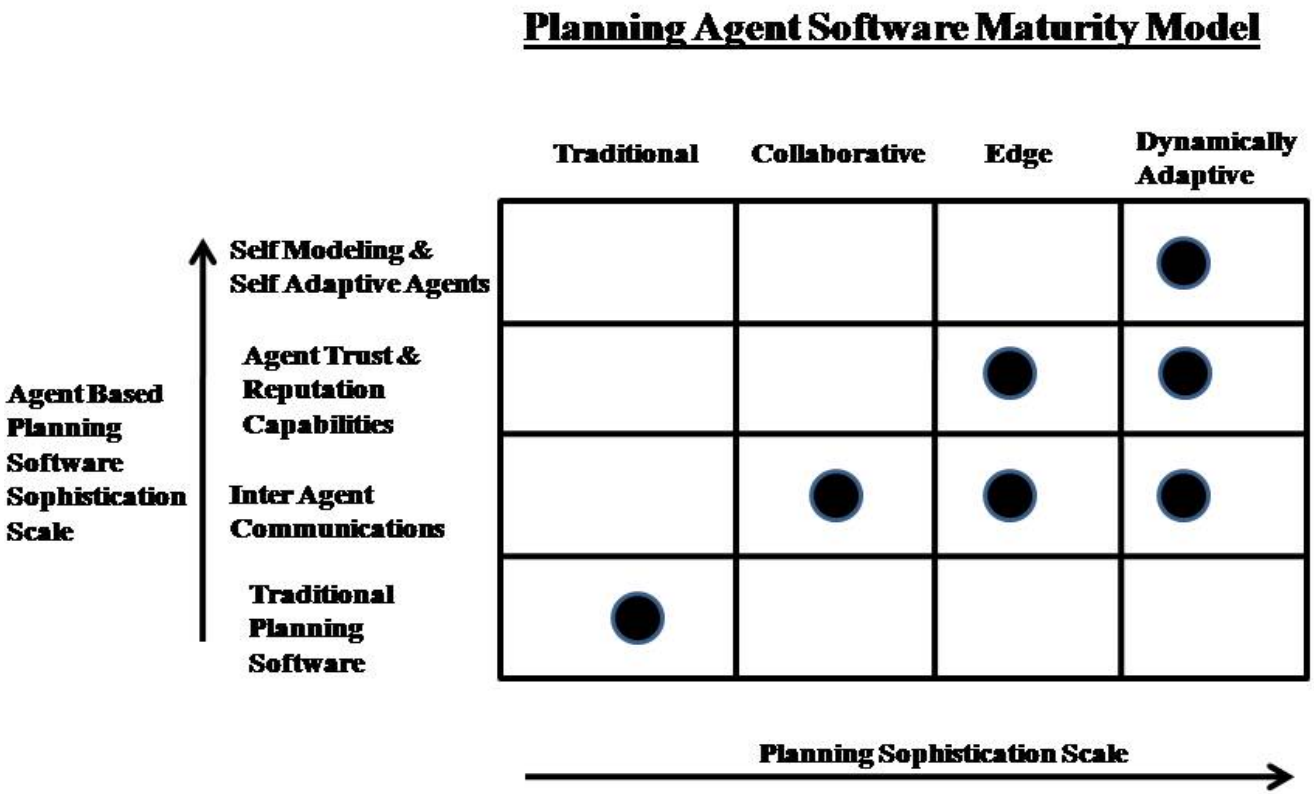
# Am I good enough at Planning?

- And if my (agent) planning skills do not match the planning tasks to be accomplished in the time predicted, do I have enough time to reconfigure myself into a better planner? (Predictive deliberation with self adaptation of the agents). By requesting services from other agents, the incomplete agent could integrate the components within its own code body or establish a collaborative planning capability with other agents with the newly required skills.
- Several schemes are available:
  - The component centric work of Karsai et.al involves a technique which includes a supervisory layer and a run time infrastructure which does not use CORBA style interfaces.
  - The next type of agent planning model which I have termed a Mogul, is based upon Minsky's[7] work and in effect becomes a "society of planning agents", each agent providing a specific planning capability invoked as required by a supervisory agent or Mogul. It is the Mogul's task to engage in contract and QoS negotiations with the other agents required for a particular task and to establish the agent based communications and data strategy required to be successful with such an architecture. The predictive deliberator in this case must have predicted that sufficient time exists for agent architecture creation on the fly, contract negotiations, and architectural testing.
  - The "Huge Local Component Library" concept usually is applied to traditional software applications. Everything in the library is put in the include file along with an enormous set of "hard wired rules" which invoke the different modules upon the demand of some supervisory component, recompiling only the required modules at run time.
  - Probabilistic agent architectures which, given a short time to reconfigure by the predictive deliberator, compute a probabilistic model of which other components or agents will be likely satisfy a given set of new requirements and then construct the probabilistic model under the assumption that the new configuration will actually generate an acceptable plan and then execute the plan at run time with no testing.
  - Workflow or BPEL based architectures, which may consist of agents, components, web services, or any combination of these structures. Architectures of this type are less inviting since they usually rely only on web service descriptions (UDDI level descriptions), or component descriptions or other types of Meta-data which do not support the rather detailed level of data modeling required to perform dynamic reconfigurability. The other issue with this approach is that the BPEL and Workflow engines are usually human centric and therefore are incompatible with dynamic reconfigurability.

# How do you get there from a software perspective?

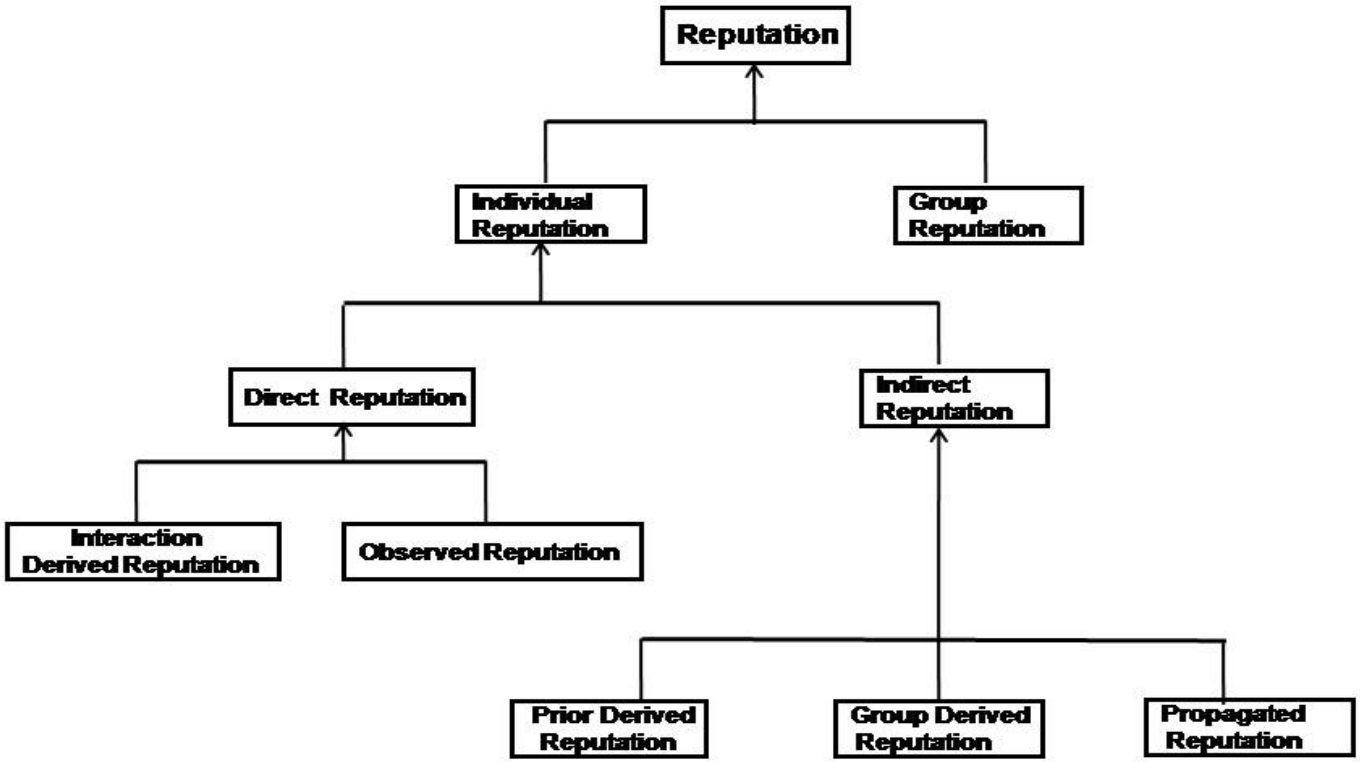


# Planning versus software maturity




Please note that for this discussion edge planning requires that decision rights be moved further down the traditional chain of command such that edge planners may execute their plans with a minimum of organizational overhead . Thus, planning agent reputations will need to emerge which will enable growing confidence in the delegation of decision rights to the edge.

# Reputation Model



# Conclusion

- In order to achieve the goals of dynamically adaptive planning and predictive deliberation, we believe that a movement must be made towards software that is able to reconfigure itself on the fly.
- Our research indicates that self modeling and self adapting agents, which can earn a reputation, offer the best approach to solving this problem.
- Our research and that of others also indicates that the use of peer to peer agent communications offers a much more flexible methodology in terms of orchestrating mission sequences than does a BPEL or Workflow engine which requires human intervention.